Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа №1**

**«Распараллеливание алгоритма вычисления произведения двух матриц»**

**по курсу: «Разработка параллельных и распределенных программ»**

Выполнил:

Студент группы ИУ9-52Б

Марченко А.И.

Проверил:

Царев А.С.

**1. Цель работы**

Распараллеливание алгоритма вычисления произведения двух матриц.

**2. Реализация**

Код программы:

#include <iostream>  
#include <random>  
#include <thread>  
#include <chrono>  
#include <vector>  
#include <ctime>  
#include <mutex>  
  
std::mutex mtx;  
  
void mult(const std::vector <std::vector <int> >& a,  
 const std::vector <std::vector <int> >& b,  
 std::vector <std::vector <int> >& c,  
 int x1, int y1, int x2, int y2){  
   
 int t = 0;  
 for (int i = x1; i <= x2; i++){  
 for (int j = y1; j <= y2; j++){  
 for (int k = 0; k < a.size(); k++){  
 t += a[i][k] \* b[k][j];  
 }  
 c[i][j] = t;  
 t = 0;  
 }  
 }  
  
}  
  
void fillMatrixWithRandom(std::vector <std::vector <int> >& mt) {  
 std::random\_device rd;  
 std::mt19937 gen(rd());  
 for (int i = 0; i < mt.size(); i++){  
 for (int j = 0; j < mt[0].size(); j++){  
 mt[i][j] = 1 + gen() % 50;  
 }  
 }  
}  
  
void printMatrix(const std::vector <std::vector <int> >& mt){  
 for (int i = 0; i < mt.size(); i++){  
 for (int j = 0; j < mt[0].size(); j++){  
 std::cout << mt[i][j] << " ";  
 }  
 std::cout << std::endl;  
 }  
}  
  
bool compareMatrix(const std::vector <std::vector <int> >& a, const std::vector <std::vector <int> >& b) {  
 for (int i = 0; i < a.size(); i++){  
 for (int j = 0; j < a[0].size(); j++){  
 if (a[i][j] != b[i][j])  
 return false;  
 }  
 }  
 return true;  
}  
  
int main() {  
  
 const int matrix\_size = 1000;  
  
 std::vector <std::vector <int> > a(matrix\_size, std::vector <int> (matrix\_size));  
 std::vector <std::vector <int> > b(matrix\_size, std::vector <int> (matrix\_size));  
 std::vector <std::vector <int> > c(matrix\_size, std::vector <int> (matrix\_size));  
 std::vector <std::vector <int> > d(matrix\_size, std::vector <int> (matrix\_size));  
  
 fillMatrixWithRandom(a);  
 fillMatrixWithRandom(b);  
  
 //printMatrix(a);  
 //printMatrix(b);  
  
 auto go = std::chrono::high\_resolution\_clock::now();  
 mult(a, b, c, 0, 0, a.size()-1, a.size()-1);  
 auto finish = std::chrono::high\_resolution\_clock::now();  
  
 auto duration = duration\_cast<std::chrono::microseconds>(finish - go);  
 std::cout << "Mult without threads: "<< duration.count() << " ms" << std::endl;  
 //printMatrix(c);  
 std::cout << std::endl;  
  
 std::vector <std::thread> thvec;  
 int max\_th\_count = 100;  
  
 for (int th\_num = 1; th\_num <= max\_th\_count; th\_num++) {  
 go = std::chrono::high\_resolution\_clock::now();  
 int chunk\_size = matrix\_size / th\_num, bonus = matrix\_size - chunk\_size \* th\_num;  
  
 for (int start = 0, end = chunk\_size; start < matrix\_size; start = end, end = start + chunk\_size){  
 if (bonus) {  
 end++;  
 bonus--;  
 }  
 thvec.emplace\_back(mult, std::ref(a), std::ref(b), std::ref(d), start, 0,  
 end - 1, matrix\_size - 1);  
 }  
 for (auto &t : thvec) {  
 t.join();  
 }  
 finish = std::chrono::high\_resolution\_clock::now();  
 std::cout << "Equal? " << compareMatrix(c, d) << std::endl;  
 thvec.clear();  
 for (auto &v: d) {  
 std::fill(v.begin(), v.end(), 0);  
 }  
  
 duration = duration\_cast <std::chrono::microseconds> (finish - go);  
 std::cout << "Mult with " << th\_num << " threads: "<< duration.count() << " ms" << std::endl;  
  
 }  
 return 0;  
}

**3. Характеристики устройства:**

AMD Ryzen 5 4600HS

6 ядер, 12 потоков

**4. Время работы:**

Mult with 1 threads: 8.101754 s

Mult with 2 threads: 3.920686 s

Mult with 3 threads: 2.945925 s

Mult with 4 threads: 2.522187 s

Mult with 5 threads: 2.231417 s

Mult with 6 threads: 2.005606 s

Mult with 7 threads: 1.830459 s

Mult with 8 threads: 1.739580 s

Mult with 9 threads: 1.666124 s

Mult with 10 threads: 1.595110 s

График: